

KAOS

For People Who Have Got Smart

President...Ron Cork
Secretary ..John Whitehead
Membership..Ralph Hess.....
Editor.....Brian Busby....
Hardware....David Anear....
Software....Jeff Rae.....
Education...Jeff Kerry.....

ADDRESS ALL CORRESPONDANCE TO :-
SECRETARY,

AIM-OSI-APPLE-MAC-IBM-RABBLE 65-C64-VIC20-PEACH-UK101-BBC

REGISTERED BY AUSTRALIA POST
PUBLICATION No. VBG4212

Vol.5 No.12

September 1985

We have had no more KAOS headers submitted this month! Does this mean you are all satisfied with what you've seen so far? Let me know at the club meeting what you think.

We want to advertise the club more widely and this can be done on various public notice boards such as the local library, computer shops, and at your work. If you are willing to distribute some copies of KAOS to the person in charge of the notice board (a personal contact is most desirable), you can pick up some extra copies each month at the meeting.

Next month, we will publish a list of club members, knowledgeable about different machines or languages, who will be able to answer technical queries or give advice.

The Junk Sale has had to be postponed since it clashed with our session with the school children at 10am on the 29th September at Essendon Primary School, Raleigh St. The school will be open at 9.30am to allow time to set up your machines. All other members and visitors will be welcome at 1pm.

The next meeting of the OSUG, Queensland user group will be on Sunday, September 29th, on the 3rd floor, Sherwood House, Sherwood Road, Toowong, starting at 12 noon.

The deadline for Newsletter articles etc. is Friday 11th October, but better to bring them to the meeting!

Have you heard?....Microsoft is rumoured to soon release a BASIC compiler for the Mac....An MPU controlled toaster to ensure your toast is evenly and reproducibly browned is now available....64K RAMs are 80c in the US.

INDEX

Adventures and the OSI	5	Membership Form	15
CP/M vs 65D - CP/M File	11	Novix NC4000 - Forth Engine	4
D/A Converter - simple	10	OSI POKE 0,X	13
Data Separator - Rabblings	14	Question Time	13
Forth Course	3	Subscriptions - KAOS	2
Index to Vol.5	14	SUPERBOARD	5
Meeting - KAOS	2	TV B&W - converting	8

THE MEETING WAS KAOS

=====

In fact it was not only KAOS, it was pretty crowded too. Probably because it was the annual general meeting and election of office-bearers. I was tempted to describe the result in historical metaphor, but then decided that I didn't want to suffer a mediaeval fate.

The election of Ron Cork as president and treasurer and John Whitehead as secretary solved the major problems. For a detailed who's who, read the header on this magazine. Jeff Rae has dropped out of active participation for personal reasons, and we all thank him and his wife for their efforts and help in the past.

Ray Gardiner brought down production units of his U.I.I.C. board, and also announced that the FORTH course would be held at Footscray the 5th & 6th of October. I'm in it for both, are you?

As we have now widened our horizons to include all genuine hardware-oriented enthusiasts (and software too, of course) (I avoided the term 'hacker' as it now has an off meaning), we will be welcoming a lot of the members from the Hampton Hackers. We understand that their meetings were deteriorating into mass copy sessions, so members of Kaos, be warned. Actually, there is safety in diversity, as the more variety we have present by way of computer types, the less mass copying there is likely to be. At any rate, H.H.'s, welcome to the KAOS. Bye now.

KAOS members subscriptions

Due to increased printing and postage cost, we are increasing the annual sub to Australia \$20. N.Z. & P.N.G. \$A25. Other countries \$A30.

Also, we are altering the club financial year to start in October and end in September. This is to line up the subscriptions with the Vol. number of the magazine and to avoid the holiday period.

As most existing members have paid up to January 1986, their subscription can be renewed any time from now until December 1985 and will be for a reduced amount (9 months) -

Australia \$15. N.Z. & P.N.G. \$A20. Other countries \$A25.

The printing and posting of KAOS newsletter is where most of subscription money is spent, as we get our meeting place free in exchange for helping the school children.

We have been having a membership drive which has been promising so far. This will reduce the cost per copy of the newsletter and delay further subscription rises.

If you have any items of hardware or software for sale that will be available for the next 12 months, EPROMS etc. as suggested in KAOS March 84, send details with your subscription.

Please renew your subscription early.

Secretary.

FORTH COURSE

REAL TIME COMPUTER BASED PROCESS CONTROL SYSTEMS

**LOCATION :- WESTERN APPLIED COMPUTERS
180 VICTORIA STREET FOOTSCRAY**

**DATE : SATURDAY 5th OCTOBER 1985 1:30 to 5:30
SUNDAY 6th OCTOBER 9:30 to 5:30**

The course will be conducted by Ray Gardiner, and is aimed at those who are planning to use the Rabble Forth board for real time control applications. A Macintosh computer and a Rabble Forth development system will be made available for each participant in the course. Much of the emphasis during the course will be placed on practical examples and interactive "hands on" experience in program development techniques

The course will cover the following subject areas,

- 1.....INTERFACING TO RELAYS, SWITCHES
- 2.....INTERFACING TO A/D, D/A CONVERTERS
- 3.....INTERRUPT HANDLING AND MULTITASKING
- 4.....TARGET COMPILING FOR ROM APPLICATIONS
- 5.....SOFTWARE DESIGN CONCEPTS (FORTH)
- 6.....INTRODUCING THE NC4000

The course assumes no prior knowledge of Forth however some knowledge of computer hardware and programming concepts will be an advantage. Due to space and equipment limitations the number of participants that can be accomodated is limited, be early!

REGISTRATION FORM

NAME : _____

ADDRESS _____

PHONE NUMBER FOR CONFIRMATION OF REGISTRATION _____

Enclosed cheque for \$28-00 course fee ☐

Make cheques payable to :- Rabble Ozi Computers
104 Macintosh Street
Shepparton 3630

BACKGROUND

The Forth language has been traditionally implemented as a virtual machine written in assembly language for the processor concerned. In this form Forth has been written for almost every CPU in existence.

The hot subject in Forth literature for the past 5 years or so has been the possibility of implementing the Forth language directly in hardware. This approach has, in the past usually meant a bit slice 2900 machine or similar construction with the Forth instruction set written in microcode. Now we have a machine in the NOVIX NC4000 that implements Forth directly in hardware. Each Forth instruction drives the hardware directly and there is NO MICROCODE, the machine is said to be externally microcoded. The Forth program is the microcode, now the real elegance of Forth can be measured in terms of direct comparison with other conventional CPU architectures. The NOVIX machine has been designed by Charles Moore the inventor of the Forth language. The actual hardware is incredibly simple when compared to conventional microprocessors.

***** Interesting fact number 1.

THE NC4000 RUNS HIGH LEVEL FORTH OVER 10 TIMES FASTER THAN THE 68000 RUNS ASSEMBLY LANGUAGE.

***** Interesting fact number 2.

THE NC4000 EXECUTES A SUBROUTINE CALL (AND RETURN) IN ONE CLOCK CYCLE. (125ns at 8Mhz). THIS IS ABOUT 30 TIMES FASTER THAN THE 8086.

Actually the subroutine (procedure call) timing will be considerably faster than just 30 times faster. Any "high" level language such as C or PASCAL will have considerable additional overhead for passing arguments (parameters) to subroutines. The advantage is probably more like 100-200 times faster than, say a 68000 application that has been programmed in C.

BASIC OUTLINE OF THE NC4000

The NC4000 has separate busses for the parameter and return stacks as well as main memory and I/O. A hardware instruction sequencer (NEXT) pre-determines the address of the next instruction within the first half of each clock cycle thus the CPU never has to wait for an address to be fetched for branch instructions. Each bit in the 16 bit word directly drives the internal hardware and one bit is reserved for ";" (the Forth return from subroutine). This allows the last instruction in a subroutine to have the RTS embedded into the instruction itself, consequently a subroutine call has an overhead of only one cycle. The internal I/O bus can be programmed in a variety of configurations, one of the features is the ability to perform comparisons in hardware rather than software.

The NC4000 represents the first of a new breed of faster, superior general purpose computers. We are about to witness the death of "conventional" cpu architecture. And the start of an exciting new era in computer architecture.

ADVENTURES AND THE OSI - Continued....

As we left off last month, we discussed the solving of adventures. This month I'll progress to writing adventures.

Why write adventures? Well, it will be largely for your own satisfaction. Just as some people enjoy the challenge of sporting competition, or climbing mountains, others enjoy programming. In the OSI world, there is little to be gained in the monetary sense. When you have completed a program that is 8k, 16k, or even 32k long, there will be a considerable sense of achievement. That will be enhanced greatly when someone who has played the game writes to you to say how much they enjoyed playing it.

One does not need a high level of programming skill to write adventures. Most of the complex mathematical functions the computer can do will go unused. You won't need to know any machine code, or fancy graphics either. The main requirement is a thorough knowledge of string handling, and an active imagination.

Don't start by sitting in front of a blank screen, and thinking "how can I write 8k of adventure". You won't have a clue where to start! The first thing to do is to have a look at as many other adventure themes as you can find. If you play adventures, you'll pick up lots of ideas for your own game. Try writing down individual scenes that have impressed you from those games. The LAST thing you should be thinking about at this stage is the programming.

The Scenario:

This is most definitely the hardest part of adventure writing. You need to outline the story that you're going to use as the backbone of the game. The plot should flow smoothly from one room to the next, with no totally unexpected events. If the impossible is about to happen, then there should be a warning, and it should not be able to kill your character. Magic has a place in adventures, but only on a very minor level. Vanishing tricks are tolerable, and the finding of a staff or rod should be enough to warn the gamer to start waving it over everything when a problem arises. There should be few random elements in an adventure, but if they are to be included, the probability should be kept low. Too many occurrences of goblins spiriting away your hard gotten treasures will frustrate you immensely.

Dungeons and Dragons games, Science Fiction novels, even ancient myths can provide the fundamental idea to get you started. It might seem that just about every possible theme has been exploited, including exploring crypts and caves, jungles, castles, outer space and underwater, but people still manage to come up with new books each year. Many less adventures have been written than novels! How about a Sherlock Holmes adventure, or one with a Wild West theme?

Hazards:

Adventures are all about encountering problems and solving them. Most OSI 8k adventures are jam packed with hazards. The adventurer never gets a moment's peace. However longer adventures, which I favour, generally let a player have a short time at the start of a game in which to get the lay of the land, find a few useful objects, and start a map, before the problems start coming in deadly earnest.

A very important consideration for the programmer is to make sure that once a hazard is dealt with, it does not re-appear later in the game. This is generally a matter of discipline in the manipulation of variables.

Map:

Having established the basic plot, and armed with your list of possible hazards, the next step is to set about drawing a map. This is even more essential than for use in solving adventures!

S U P E R B O A R D

Make sure that you have plenty of paper. Then draw in the boxes that represent each room, and pencil in the hazards and room objects. Much re-drafting is generally the order of the day at this early stage. Gradually, the map will be refined until you are satisfied, and at this time you will have a very firm grasp of the plot. You will know what will happen to the player and just where it will happen. Your (almost) final map will also contain the goodies needed to solve the adventure. The order of objects must be such that sufficient are encountered so that each hazard can be solved before moving on to the next stage. There isn't much use having a key where it cannot be retrieved before trying to open a door. An unsolvable adventure is quite unpardonable!

Programming:

At last! The start of the endeavour! The first step in programming is to put in the DATA statements and read them into variables. All the room names, directions, and object names. Having done this, you pencil in the variable names next to the object names, and of course, the variable subscripts, and number the rooms. Every adventure has a list of nouns and verbs. Usually the entire word is not used, but only the first two or three letters of each word. The result looks like this in the program: V\$=COGESHKILOOPDRUNLIRETIBR. It is most important that verbs or objects don't have the same two letters, or the program won't find the second one. If you want to use DROP and DRINK as verbs, then it is best to use three letters, or find an alternate word. If there are three doors in the adventure, rename two of them to gate, barrier, portal - or anything else you can think of. There are ways around having several objects with the same name in different rooms, but it is to be avoided if possible. You will also probably have to add a few objects you didn't first think of, when you get around to the debugging stage, but more of that later. The verb list will be dictated by what needs to be done to solve problems. You should also try to think of other verbs a player might use, and redirect them to your favoured verb. Too many responses of "I don't understand TA" can be the cause of a loss of interest in the game. With 8k games, however, there is little room for frills.

Mazes:

Most larger adventures have some form of maze, generally in the form of a number of rooms with the same name. Hard mazes are easy to construct, and can be frustrating to negotiate. What usually happens is that if you don't follow the correct path, you go back to the first room. Of course it has the same name as the others so you won't know that! The solution to an adventurer trapped in a maze is the same as the one you would use in real life. Mark the trail! Once having mapped the maze, you can pick up the objects you drop. Care must be taken to ensure that what you drop won't break, else the adventure might just become unsolvable.

Movement:

Most adventures give you the obvious exits. Of course not all exits may be obvious! However, the original Adventure didn't. You were left to find them for yourself. A map in this situation is an absolute necessity. I don't like the latter approach myself. Too many "You can't go that way" messages tend to make me lose interest. A middle road might be to make hints in the text, but as OSI memory is restricted usually, I think the first option is the best.

Responses:

This is one of the most important factors in keeping the interest in the game high. It is imperative that you try to estimate the responses of the player to your problems. Everyone is different, and you could waste a lot of code on responses that will never be used, but you should at least try to estimate likely responses. Some different answers, selected at random, might keep it interesting. Also use of the word "yet" at the end of a refusal can keep the player trying.

S U P E R B O A R D

```

100 GOSUB999:PRINT"Vampire's Treasure":PRINT:PRINT
110 DIMR$(11),N$(32),L(32),C(9,6)
120 FORR=1TO8:READR$(R):NEXT:V$="GOTALOOPDRUNLIRESHTIKIBRGE"
130 FORR=1TO32:READN$(R):N$=N$+LEFT$(N$(R),2):NEXT
140 FORR=1TO32:READL(R):NEXT:FORR=1TO9:FORY=1TO6
150 READC(R,Y)NEXTY,R:H=1:R$(9)=R$(6):R$(4)=R$(6):GOSUB999:GOTO950
199 REM Main Loop
200 PRINT:INPUT"What now";A$:PRINT:B$="":N=0:V=0:IFLEN(A$)<2GOTO280
210 C$=LEFT$(A$,2):IFC$="IN"GOTO900
220 FORR=1TOLEN(A$):IFMID$(A$,R,1)=" "THENB$=MID$(A$,R+1,2)
230 NEXT:FORR=1TOLEN(N$)STEP2:IFMID$(N$,R,2)=B$THENN=(R+1)/2
250 NEXT:FORR=1TOLEN(V$)STEP2:IFMID$(V$,R,2)=C$THENV=(R+1)/2
260 NEXT:PRINT
270 ONVGOTO300,350,400,550,500,550,600,650,800,750,800,850,350
280 PRINT"I don't understand you":GOTO200

950 REM Location:RETURN

999 FORR=1TO32:PRINT:NEXT:RETURN
1000 DATAsmall chamber,sloping cave,high walled garden,tunnel
1010 DATAlarge cavern,diamond mine,vampire's lair,huge vault
1020 DATALANTERN,MATCHES,SKELETON,GUN,ROPE,GRASS,FLOWERS,STAKES,KEY
1030 DATABULLET,DYNAMITE STICK,BONES,DIAMONDS,SIGN,TUNNEL,SCULPTURE
1040 DATAMIRROR-WALL LENGTH,HOLE IN THE FLOOR,URN,ELEPHANT,VAMPIRE
1050 DATABAT ON HIGH SIGNPOST,GARDEN-OVERGROWN,IRON DOOR-MASSIVE
1060 DATAENTRANCE-SEALED,LOCK,NORTH,EAST,SOUTH,WEST,UP,DOWN
1070 DATA1,0,-2,0,5,3,3,0,9,0,6,0,8,0,0,0
1080 DATA-1,-4,0,-2,-7,-5,-3,-2,-7,-8,0,0,0,0,0,0
1090 DATA0,0,2,0,0,0,1,0,0,0,0,0,5,0,0,0,0,6,1,0,5,0,7
2000 DATA0,4,3,0,0,0,6,9,4,6,0,0,0,0,0,0,4,0,7,0,0,0,0,9,9,9,6,0,0

```

The program that appears above is the bare bones of the adventure. The reading of the data into variables was covered in *Treasure Quest*, so I won't repeat it. In line 120, you find a strange string. V\$ consists of the first two letters of all the verbs used in the adventure. Line 130 does a similar thing for the nouns, forming N\$

Lines 200 to 280 form the program loop. The whole adventure is controlled by the string handling routines within. The answer to the "What now" question is expected to be a verb and a noun, or a verb or noun only. The only noun handled is INVENTORY. This will remind you of what you are carrying.

Lines 220 to 260 breaks up your answer into the first two letters of the first word, which becomes C\$, and the first two letters of the last word, if any, which becomes B\$. The words are expected to be separated by a space. The routine will accept LO SI or even LOOK AT THE SIGN. If your answer was shorter than two letters, or if the verb is not found in V\$, you get a "I don't understand" message, and are passed back to the input again.

Line 230 goes through the noun string, searching for a match to B\$, the noun part of your answer. If it matches, the variable N contains the position of the noun in the string. L(N) will give the room number for the noun. If L(N) is 0, you can't see it. If it is negative, you can't take or get it. Line 250 does the same thing for the verbs. Depending on the position of the verb in V\$, line 270 will transfer control to the appropriate routine to handle that verb. The rest of the adventure consists of individual verb/noun handling routines.

If you want to have a go at writing your own adventure, or even converting one from one of the more popular makes of computer, an understanding of the above routines will make this task considerably easier.

Next month, the rest of the adventure, and a most useful marriage for WP-6502.

Ed Richardson.

CONVERTING A MODERN 12" B & W PORTABLE T.V. for computer usage
i.e. video & audio input -- by Wolf.Horn

In the good old days when 12" portables had discrete circuitry, you could feed the computer video output through a 10-22 uFD Tantalum Capacitor (high z) and tap in just after the video detector stage, turn off the supply voltage (B+) to the tuner and away you went. Audio was and still is easy. Just break in at the top of the volume control using part of your switch that switches from monitor to T.V. mode. (using shielded cable of course).

The advantages of using a normal T.V. set are of course that it can be used for 2 purposes. I personally prefer B & W T.V. sets as an alternative for games etc. when the colour set is being used by the boss because they provide you with a better contrast range than say a green screen can. The green or amber screens are the best for text processing of course.

My first B. & W. 12" monitor/tv set was 12 years old and not worth repairing. So I bought myself a \$99 General T.V. & the \$4.95 Dick Smith Modulator but no matter what I tried the result was unsatisfactory. Consequently I sold it and bought a National B. & W. 12" model TR-602AA (chassis 12G02-U) as well as the better modulator from Dick Smith Electronics. This unit is called (get ready for it) 'VHF INTERCARRIER VESTIGIAL SIDEBAND MODULATOR' cat.no. K-6043 at \$ 9.95 and well worth the money spend I might add.

A WORD OF WARNING !!!! - before you hardwire this unit into your T.V. set, try it out on a temporary basis first since some of the T.V. sets on the market are just not worth converting.

PARTS NEEDED:

- 1 VHF (CH.1 or 0) R.F. MODULATOR
- 1 R.C.A. Plug
- 2 R.C.A. panel sockets
- 1 only 2 pole, single or double throw switch
- plus some hookup wire, shielded cable for audio I/P and 75 ohm coaxial cable for the video and R.F. connections to modulator.
- 1 150 ohm 1/2 watt carbon resistor
- 1 220 uFD, 250V electrolytic capacitor
- 1 0.1 uFD greencap (polyester) bypass capacitor

The R.F. Modulator has connections as shown in diagram 1.

Wire R.C.A. Plug to the 75 ohm Coax. and plug into modulator. Connect the free end of this wire to the input terminal or P.W. Board track that goes to the tuner input. (Be careful and make absolutely sure you've got the tuner input track and 'NOT' the earthy side). Connect the braid of this lead to the tuner earth terminal or track. Connect a lead from modulator case (=earth) to the earth terminal/track on the tuner P.W. board.

The +VE side of the 11.5V supply voltage is connected as per diagram 2, using the 150 ohm resistor and the 2 capacitors to drop the supply voltage from 11.5V to 6.5V and provide some additional filtering (to stop interference).

Leave the channel selector wire free, just shorten & insulate it (for safety). This will select Ch.1 VHF. To select Ch.0, you solder this wire to the case (earth) of the modulator.

Connect some 75 ohm coax. to the video terminal (Pin 1) of the modulator with the shield going to the case and the free end of this cable going to the R.C.A. panel socket.

Keep the audio and video input cables (to modulator) as short as possible. The best place to mount them is on the flat portion of top back section of the portable. Run the audio cable in the same way using shielded cable. Don't forget to label the sockets. The switch is mounted on the antenna terminal pads. One half is used to switch the 11.5V supply and the other half can be used to short out the incoming antenna signal at the terminals as is shown in diagram 3. NOTE:- you may not have to short out the antenna terminals, but if the picture is not steady and touching the telescopic antenna makes it worth then this extra mod. will provide you with a rock-steady picture.

Last but not least check your wiring and switch it on with video and audio coming from your computer. Select channel 1 on your T.V. & adjust the fine tuning for best picture and sound. NOTE:- you may have to carefully tune the 2 slugs (coils) on the modulator to get a good picture and good sound !!

Building the modulator into the T.V. provides a superior result and it also lets you use several computers. (e.g. if you change computers you don't have to buy a new V.H.F. modulator. At the same time don't think for one minute that this mod. will fix every problem but the problems I had and the subsequent cure sure made me happy since I obtained a near perfect result.

Diagram 1

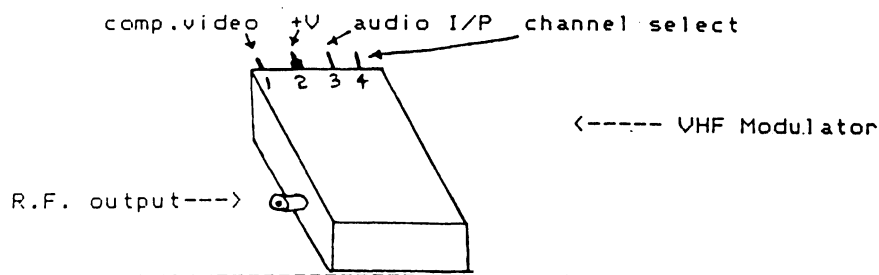


diagram 2

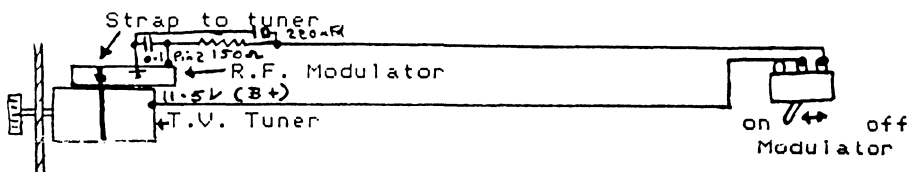
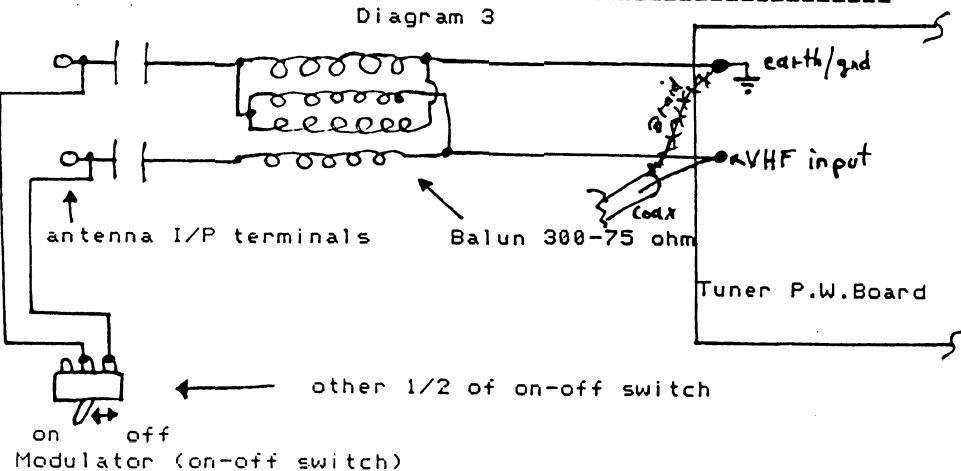


Diagram 3





SIMPLE D/A CONVERTER

A relatively simple way of achieving D to A conversion is to use an op amp as the device and simply alter the voltages to its input to affect the analog output. This is achieved by using the correct value of resistors on the input of the LM741 to 'weight' the input according to the binary input applied.

No one would claim that this circuit, with only four inputs, is as effective as the professional devices but it is cheap and easy to build and demonstrates clearly the principle involved.

To switch this D/A converter it is necessary to apply a 'high' to one or more outputs of the 'simple I/O port' project. This is done by poking to the games port in the following manner.

RESULT	PROGRAM LINE
ENSURES ALL OFF	5 POKE 16295,0:POKE 16293,0
	10 POKE 16291,0:POKE 16289,0
(1)	15 POKE 16290,1
	20 GOSUB 1000
	30 POKE 16289,0
(2)	40 POKE 16292,1
	50 GOSUB 1000
	60 POKE 16291,0
(3)	70 POKE 16290,1:POKE 16292,1
	80 GOSUB 1000
	90 POKE 16289,0:POKE 16291,0
(4)	100 POKE 16294,1
	110 GOSUB 1000
	120 POKE 16293,0
	1
	1
	1
TIME DELAY	1000 FOR X=1 TO 2000:NEXT X

This can be continued until the binary equivalent of 15 is reached (1111) I.E. all outputs turned on. This will cause the output of the D/A converter to rise in a series of steps towards its top voltage.

The steps are nowhere near as smooth as an 8 bit input of course, because with 8 inputs to play with it gives a total of 255 (11111111) voltage steps and so is much smoother.

Because of space limitations, I have not included the full program to test this project but the above should give you the idea of poking the games port on and off in a binary arrangement to suit your purpose.

The values below may help you calculate the combination required for any given value.

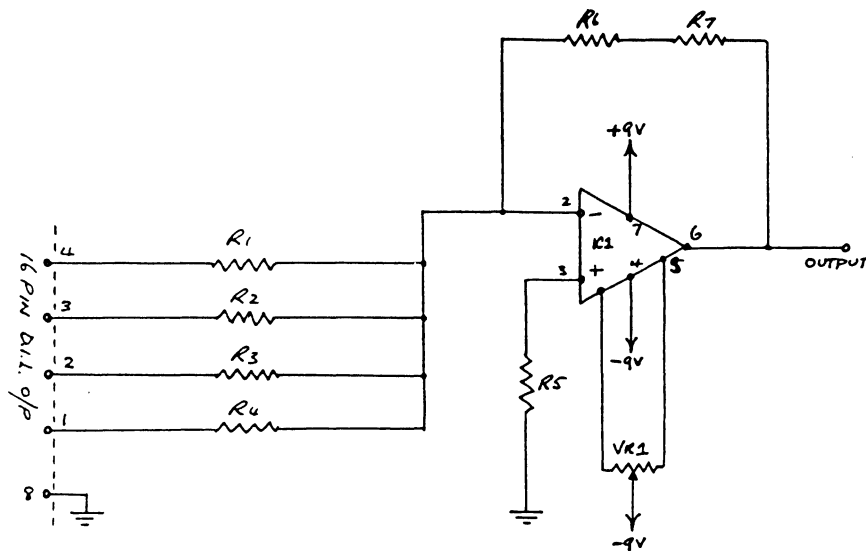
16290,1 = 1
 16292,1 = 2
 16294,1 = 4
 16296,1 = 8

16289,0 = off
 16291,0 = off
 16293,0 = off
 16295,0 = off

NOTE: All resistors should be 1% if you want maximum accuracy from the output, but the nearest value 5% resistor may be used if you only wish to demonstrate the principles involved.

SIMPLE D/A CONVERTER.

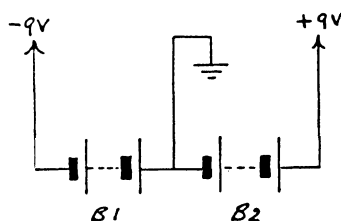
BY R.WOODHOUSE.



PARTS LIST

IC1	LM741	VR1	10K
R1	100R	R2	390R
R3	1K6	R4	6K2
R5	75R	R6, R7	100R
B1, B2	9V transistor battery		

All resistors to be 1/2 watt, 1%.



THE CPM FILE

=====

I promised to do a comparison on OSI [65D + Compdos 1.3] and CPM with particular reference to their operation on the Rabble, so here goes. I do not vouch for the accuracy of every single statement as some of this is second-hand, and a lot is in any case a matter of interpretation and/or opinion. No correspondence will be entered into - this is my final decision (like, I don't have the time).

Both operating systems in fact have a lot in common, but they are quite different to use. Let me first try to describe the OSI O.S. in about a paragraph.

The OSI system could best be described as a primitive but still very powerful O.S., especially as enhanced by CD 1.3, which actually introduces some UNIX-like (UNIX is another O.S. and another story too) features. I call it primitive because it is the operator's responsibility to look after a lot of the 'house-keeping' of the system. This of course means that it is easy to crash the system due to accidental operator error, such as forgetting which disk you are on, or some such. In terms of size, the OSI system is quite big, being nearly 16k, including CD 1.3. However, this does include the Basic that is always booted up as an integral part of the O.S., and that accounts for nearly 10k, which can be overlaid by such utilities as WP6502, SDP, AS, EM, Forth or UCSD Pascal. I guess that when you get down to it, the main deficiencies of the OSI system are:-

- the disk I/O uses nonstandard hardware (so does Apple).

- the disk capacity is small and the organisation is inefficient.

- the O.S. is really not all that 'user-friendly'.

- the choice of languages and utilities is not very large.

On the other hand it is a fairly simple and small system so that the operator can quickly build up a working relationship and get down to doing something useful.

The CPM system divides memory into four parts, which together make up the system:-

- BIOS basic I/O system (hardware dependent and in the case of the Rabble has been partly translated into 6502 code and much of it lives in ROM at A000-BFFF).

- BDOS basic disk operating system (this part is sacred).

- CCP the console command processor (this is the operator interface, which accepts typed-in commands, and puts the responses on the screen, etc. and in our case the standard CCP has been replaced by ZCPR 1.6 as this has a few extra bells and whistles).

- TPA the transient program area. This is where all the programs are loaded for execution. CPM uses page 00 to store frequently used variables and pointers, so the TPA starts at page 01 and goes up to the base of the O.S. which is loaded into the top of memory.

The O.S. (BIOS+BDOS+CCP) in CPM is normally about 13k, but in the Rabble system takes up less than 9k and sits above the work-program area (TPA) and when you load and run a program, it usually takes over from the CCP, and may even overwrite it, in which case you have to do a 'warm boot' (^C) to load a new copy of the CCP and BDOS, under the control of the BIOS. And that is as technical as I am going to get. If you want more nitty-gritty, there are many good books on the subject of CPM. Buy or borrow one.

CPM is relatively 'user-friendly' and a lot of the routine housekeeping is done automatically. It tends to be a bit back-to-front in the way some commands operate, but you get used to that. The disks hold a lot, 1368k per disk for 8", 792k for 5" 80-track and about 380k for 5" 40-track. Because it does so much more, and because the disk capacity is high and sector-based (i.e. 128 byte units) it can sometimes seem slow compared to simpler, less efficient systems, but it really is quite fast. Several hundred thousand users around the world can't all be wrong. The biggest problem you come up against when changing over from say OSI is great variety of software available, and you end up with mental indigestion. Bye now - rmh

QUESTION TIME??

OSI POKE 0,X

When ROM BASIC is cold started, it sets up page 0, 1 and 2 with pointers etc that it needs when operating. If after cold starting, you use EXMON to disassemble the start of page zero, the result is:-

BASIC in ROM	HEXDOS
0000 4C74A2 JMP \$A274	0000 4C3C03 JMP \$033C
0003 4CC3A8 JMP \$A8C3	0003 4C830A JMP \$0A83
0006 05AE ORA \$AE	0006 A506 LDA \$06
0008 C1AF CMP (\$AF,X)	0008 C1AF CMP (\$AF,X)
000A 4C88AE JMP \$AE88	000A 4C6D04 JMP \$046D
000D 00 BRK	000D 00 BRK

If you exit from BASIC by pressing BREAK, it does a hardware reset of the 6502 uP. The uP is programed internally to start running again at memory location \$FF00. The machine code at \$FF00 onwards resets the STACK, the input, output, load and save vectors, the cassette ACIA, clears the screen and prints D/C/W/M ? then waits for your command. If you reply with W, the m/code at \$FF4E is run and you jump to memory location zero.

```
FF4A C957    CMP #$57    ($57 = ASCII W)
FF4C D003    BNE $FF51
FF4E 4C0000  JMP $0000    At location zero, it finds a JMP $A274,
which warm starts BASIC in ROM.
```

Any program can make use of the Warm start after BREAK if it previously pokes JMP and its warm start address into the first three bytes of page zero. WP6502 does this as shown below, so it can be Warm started by pressing W. (my WP 6502 is in EPROM at \$8000)

The assembler does not do this, so can not be warm started with W.

WP 6502	Assembler
0000 4C0B8F JMP \$8F0B	0000 8100 STA (\$00,X)
0003 BF ???	0002 80 ???
0004 8F ???	0003 00 BRK

Has this answered your question, Bob Best ?. (KAOS July 85 page 3)

If you try to disassemble the above and find that EXMON wont disassemble \$0000, then your EXMON has a fault in it. You can either alter two instnuctions in the EXMON Q code to make it like the code below or disassemble from \$FFFF.

Original EXMON	My version
0014 20110B JSR 0B11	ED29 2023EB JSR EB23
0017 8505 STA D5	ED2C 8555 STA 55
0019 A50B LDA DB	ED2E A50B LDA DB
001B 8506 STA D6	ED30 8556 STA 56
001D 20070B JSR 0B07	ED32 8608 STX D8
0020 8608 STX D8	ED34 209CE9 JSR E99C
0022 209C09 JSR 099C	ED37 20A3E9 JSR E9A3
0025 20A309 JSR 09A3	ED3A 2019EB JSR EB19
0028 20070B JSR 0B07	ED3D 2053E8 JSR E853
002B 205308 JSR 0853	ED40 C90A CMP #0A
002E C90A CMP #0A	ED42 F0F0 BEQ ED34
0030 F0F0 BEQ 0022	ED44 4C32E8 JMP E832
0032 4C3208 JMP 0832	

John Whitehead.

GENERAL		SOFTWARE	
68000 Project	3 11	ABS Function	11 7
Braille and Computers	1 3	Adventures and the OSI	12 5
Disk Unbuckling	8 1	Adventures and the OSI	11 6
Education - Computers in	11 4	Apple	9 5
Filing System - Personal	11 2	Apple - Jackpot	8 4
IBM Compatables	5 5	Backspace fix - Serial	8 8
ROM BASIC Utilities	3 15	Binary Counter	3 14
Viatel - accessing	11 5	Biorythm Calculation	3 14
HARDWARE		C1P DOS Notes	2 2
Apple Disk II Analog Board ...	2 11	Compdos 1.2 - relocating	11 17
Apple Disk II Interface	1 14	CP/M vs 650 - CP/M File	12 11
Apple IIe - Enhanced	8 3	DOS Notes	5 10
ASM65 - Adding to SBII	8 13	EPRom Programmer	6 8
C1P in Candlelight	1 12	Fast Sort for OSI	2 4
5" Disks - CP/M File	12 6	Garbage Collector Fix	2 5
D/A Converter - simple	12 10	Graphics Characters	5 5
Data Separator - Rabblings ...	12 14	Hexdos	10 10
Disk 5.25" - modify	6 4	Keyboard - OSI	8 11
Forth Board - Rabble OZI	11 8	M/code and 6503.2	9 13
I/O Port - simple Apple	11 2	Mailing List Program	3 5
Mackintosh - The	11 12	MDMS - More On	1 12
Memory expansion - C1P	10 4	Monitor Locations - C1P	8 15
Modem - D.I.Y.	3 10	Nulls - getting more	8 8
Modem - split baud rate	10 5	OSI POKE 0,X	12 13
Modem Phone Enhancements	9 2	Random Access Files	1 4
Novix NC4000 - Forth Engine ..	12 4	Random Access Files - more ...	8 6
Parallel Printer Boards	10 5	ROM Integrity Test	1 6
RAM Add to S/B	9 4	ROM Test Expansion	3 2
ROM - Extra	3 2	Search substring	3 4
RS232 - Commodore 64	9 12	Sink the Fleet - Apple	9 8
Sound Generator - Apple	10 2	Sound Generator - Rabble e/b .	9 14
Superboard - My part 11	12 10	Statistics	3 6
TV B&W - converting	12 8	Tape Library - KAOS	10 13
REVIEW		Tape Token Format Fault	2 3
Frog Jump	9 4	Video 6545 - Update	6 6
Mouse - AMX for BBC	9 6	Visicalc - power - part 2	11 14
Nice Print - Apple i/f	9 7	Visicalc - power of	10 7
Number Matcher - Review	5 5	WP6502 Revisited - pt 2	5 11
Small Business Analysis	3 4	WP6502 V1.2 Revisited	3 17
Super Doodler - Review	1 6		

Rabblings - Data Separator
by Gerard Campbell

Recently while building the Rabble expansion board I found that one chip forming the basis of the data separator (as in the Rabble 65), namely the 8602/9602 was harder to get than a tribe of Jamacian Pigmies. More or less by accident I found that a direct replacement (pin for pin) is the CMOS 4528, readily available even from Dick Smith's !! Normally when interfacing TTL to CMOS, a pull-up resistor is placed on the TTL O/P driving the CMOS I/P. I did not do this and the drive works perfectly.

SCOOP - a new computer language will be released soon, similar to FORTH but twice as fast, twice as flexible and two times easier to learn; it is called TWOTH (TOOTH ?). This news is direct from silly-con valley. (sic)

KAOS

Secretary
John Whitehead

Membership Application.
Forward to the secretary
together with your fee.

Subscriptions run from October to the following September.

The subscription fee pays for a full years supply of the club magazine and postage.

Subscription rates : Australia \$20, N.Z. & P.N.G. \$25, Other Countries \$30.

Make cheques payable to KAOS in Australian dollars on any Australian bank.

There is no discount for mid-year entry, instead we supply all issues of the magazine for those previous months.

NAME : PHONE :

ADDRESS : NETWORK/DATABASE NAME:

..... USER-NAME :

..... AMATEUR CALL SIGN :

..... RTTY CALL SIGN :

	Computer No1	Computer No2	Computer No3
Make/Model			
Memory Size			
Cassette Baud			
Disk Size			
Density			
No of Drives			
DOS type CP/M ?			
Screen Format			
Printer type			
Modem Baud rates			
Other peripherals			
Programming skills.	BASIC M/Code FORTH FORTRAN	OTHER.....	
Other Interests.			

My name MAY/MAY NOT be included in lists circulated to other KAOS members. (Strike out whichever is not applicable)

NOTE : Lists are not circulated to non-KAOS members.

Signed : Date :

Registered by Australia Post
Publication No. VBG4212

If undelivered return to
KAOS, 24 Archibald St.
Pascoe Vale, VIC. 3044

KAOS KAOS
A Postage K
A paid A
S Essendon S
A Victoria A
A Australia A
O 3040 O
S KAOS KAOS